**CSCI 232 - Out Lab 3 (100 pts max)**

**Due: 11pm,** ~~October 23, 2020~~ **November 6, 2020 (extended)**

**Overview:**

In this lab, you will write a program to provide the core functionality of a car GPS routing system by computing shortest paths in a graph. You must implement a shortest path algorithm (Dijkstra’s algorithm).

***Note:*** *For this lab, I would like you to implement your own graph-related classes but you can use non-graph related Princeton or built-in Java classes.*

Your program should read in a graph file (the format is discussed below), create an internal representation of graph, then present the user with a menu:

The current graph has vertices from 1 to 2500.   
Would you like to:

1. Find a new route
2. Exit

If the user wants to find a new route, ask them for the *source* and *destination* (just use the vertex integer indices to specify) and then find the shortest path between them and print it, along with the total distance.

Your program should run from the command line and read the graph in from a file:

java MyGPS graph.gr

The graph.gr input file has the following format:

c 9th DIMACS Implementation Challenge: Shortest Paths

c http://www.dis.uniroma1.it/~challenge9

c TIGER/Line graph USA-road-d.NY

c

p sp 264346 733846

c graph contains 264346 nodes and 733846 arcs

c

a 1 2 803

a 2 1 803

a 3 4 158

a 4 3 158

a 5 6 774

etc.

Here ‘c’ means comment, so ignore, ‘p’ provides the graph sizes (vertices and edges) and ‘a’ means an edge with the given weight. Fully specifications are here:

<http://www.dis.uniroma1.it/challenge9/format.shtml#graph>

**Notes:**

* You should first create a small test graph with ~10 vertices and ~20 edges to test your code (you can share these in your lab for testing purposes).
* The website  
  <http://www.dis.uniroma1.it/challenge9/download.shtml>   
  has many test files. You should minimally get your program to work on the **Rome99** data set and see if you are able to handle larger data sets such as the **NY** data set.
* Calculate the computation time it takes to find the shortest path (you can use System.currentTimeMillis to get the current time).
* You are required to implement your own graph data structure and implement Dijkstra’s algorithm for finding shortest paths.

**Output:**

You output should look something like this for each query:

Shortest path from 1 to 2:

1 -> 5 -> 4 -> 3 -> 2

total distance: 4567

time to find: 0.3 sec

*Some points will be awarded for nicely-formatted output.*

**Bonus Problems: (+ 5 bonus pts each)**

Some of the test files include the lat-long coordinates of the vertices. Use the StdDraw class from the textbook to plot the shortest path in a window; you will need to figure out a way to map from lat-long coordinates to pixel locations.

**Submission:**

Submit on gradescope.com using your MSU credentials prior to the due date/time (you may also submit in-person directly to your TA during lab). Submit your:

1. .java files
2. an input file that you’ve created to test the program (but not the big graph files)
3. the resulting output and your discussion (can be one document).

*Only one submission per group is required BUT you must put ALL group member names in the comments of your program.a*